ECSE 416 – Telecommunications Networks

Experiment 2

Wireshark TCP Analysis

Fall 2020

Group 16

Aleksas Murauskas 260718389

Florence Diep 260727117

October 19th, 2020

1. Introduction

The objectives of this experiment were to analyze the behaviour of the transmission control protocol (TCP), which acts as the primary transport layer protocol to provide reliable data transfers. By using the Wireshark features to analyze TCP traces from various servers around the world, we were able to identify the different TCP congestion control phases (i.e. slow start, congestion avoidance and packet losses and gathered packet trace data to quantify the throughput of a TCP connection. The tool allowed us to capture packets and investigate the contents and performance of TCP under different download conditions.

We have four TCP packet traces that capture the congestion performance of downloading a Ubuntu Linux distribution from Australia, Kenya, Lithuania and Slovenia with duration between 2-3 minutes. Our results show that the average throughput heavily depends on the distance between the server and receiver. As the distance gets larger, a higher packet loss will occur.

2. Experimental Setup

The traces were collected by setting Wireshark to record as soon as the Ubuntu distribution is downloaded and stop recording until the desired duration prior to cancelling the download from the server. The experiments were run on a laptop in Montreal with the network connected to WiFi. The server locations used were Australia, Kenya, Lithuania and Slovenia, which were chosen based on having a country from each continent and variety in location distance from the laptop. We executed the downloads on a Wednesday morning at around 10AM. During this recording, we made sure to minimize all other web activity to minimize any network disruption, but the biggest limiting factor is the distance between the server location and the laptop. In addition, the further the server is, the more routers and networks the packets will need to pass by, which may lead to encountering congestion or other problems on their way.

3. TCP Behaviour Comparison

3.1 Australia

Below in figure 3.1.1 is our stack trace we found for Australia. It shows a fairly linear transmission, meaning that packet loss was rare and its impact on the transmission was small.

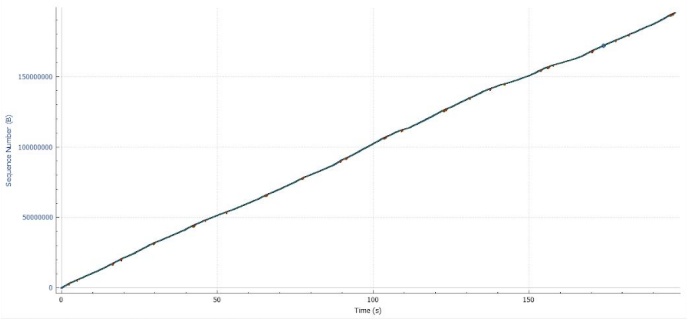

Figure 3.1.1 - Australia Sequence Numbers (TCP Trace)

In this TCP trace, we are able to identify a slow start beginning at about 8.05 seconds, which exponentially increases the window Congestion avoidance is resumed around 8.15 seconds.

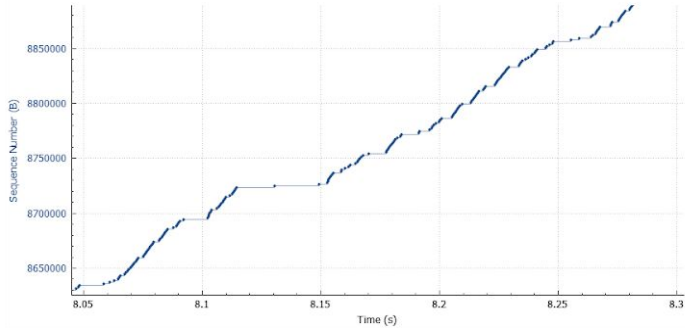

Figure 3.1.2 - Australia Slow-Start and Congestion Avoidance Captures

In figure 3.1.2, We can see the program operate through fast transmission as there are multiple packets received out of order between 42.5 and 43.2.

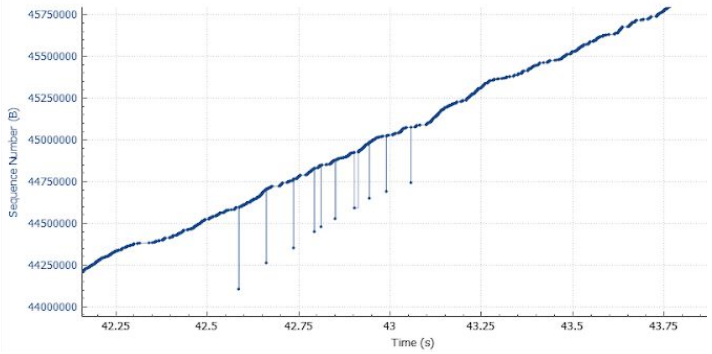

Figure 3.1.3 - Australia Fast Retransmission and Out-of-Order Captures

Below in figure 3.1.1 we can see the maximum window size is 16653 bytes. The Trace begins with a window size of 8326 bytes, but it doubles at 16.8 seconds into the capture and remains at this maximum window size for the remainder. The spikes in the figure represent times where the sender sent packets far above this window size. At these points many packet losses occur in the receiver end as the sending rate becomes faster than the rate of acknowledgement of arrival. As can be seen in Figure 3.1.4,

Average Throughput was oscillating around a throughput of about 8 million bits/second, or 1 million bytes/second.

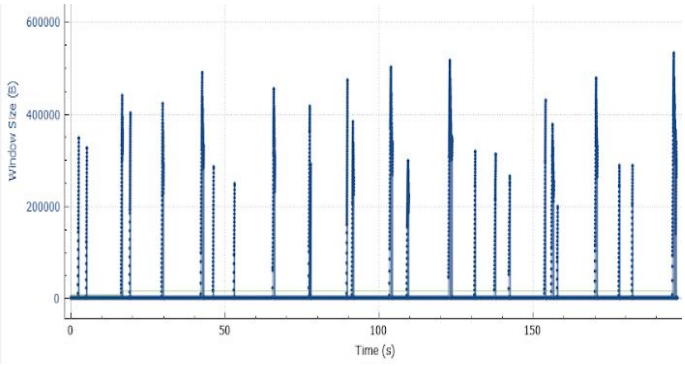

Figure 3.1.4 - Australia Window Scaling (Server to Receiver)

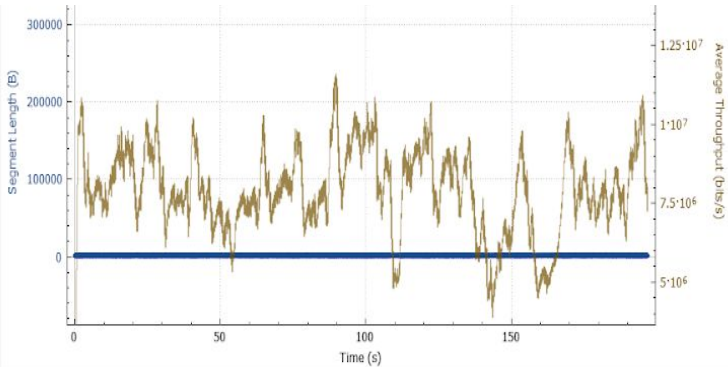

Figure 3.1.5 - Australia Throughput (bits/s) (Server to Receiver)

3.2 Kenya

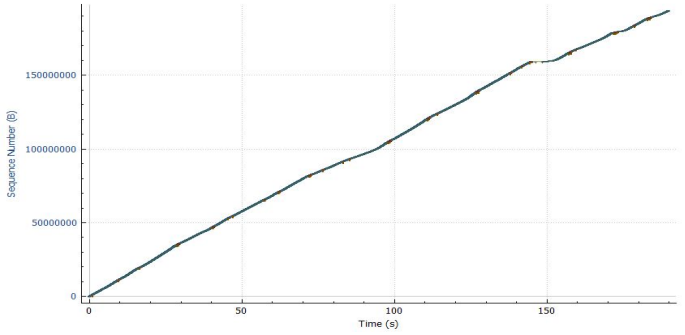

Figure 3.2.1 - Kenya Sequence Numbers (TCP Trace)

Kenya displays a TCP congestion window behaviour that easily allows us to identify the events, namely due to the frequency of packet losses. For instance, a slow-start can be clearly identified around the 150 seconds. As seen in figure 3.2.2, we can see that a slow-start has initiated after a timeout event has occurred at around 145 seconds captured by Wireshark, triggering a packet loss event. Hence, in order to recover from the lost segment, we can see that the server has broken the congestion avoidance algorithm and has significantly dropped the sending rate to retransmit the missing segment, and proceeds to resume congestion avoidance phase before encountering more packet losses at around 158 seconds.

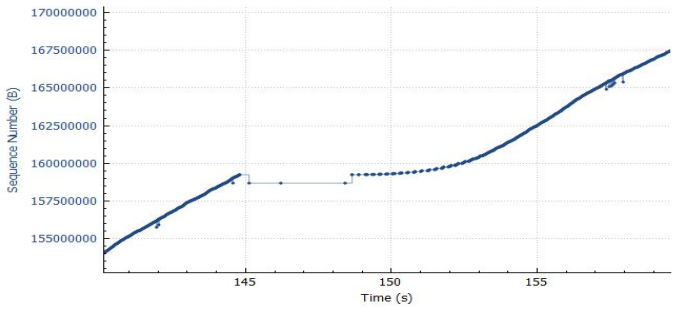

Figure 3.2.2 - Kenya Slow-Start and Congestion Avoidance Captures

In the trace, we were able to find several retransmissions and out-of-order packets. Using Wireshark allows us to assess to classify the type of loss. In figure 3.2.3, there are several fast retransmissions in a row and an out-of-order packet loss has occurred right after a retransmission at around the 29.5 second mark. In both cases, we can see that every time a packet loss occurs, the congestion-window algorithm enters the slow-start phase.

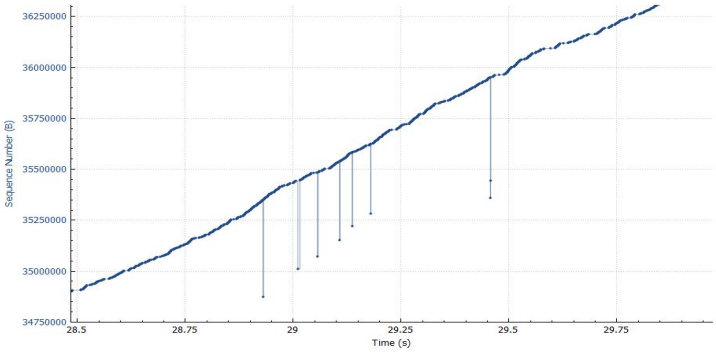

Figure 3.2.3 - Kenya Fast Retransmission and Out-of-Order Captures

In regards to the maximum congestion window sizes, we can see that the maximum congestion window size is 16516 bytes. It initially starts at 8258 bytes, doubles at around 49 seconds and remains at the value for the remainder of the trace. It is clear that the sender is sending packets beyond that specified congestion window size several times. The spikes in the TCP Window Scaling Stream graph (figure 3.2.3) show that this segment increase caused many packet losses to occur in the receiver end as the sending rate becomes faster than the acknowledgement arrival rate. This creates buffer overflow due to uneven bursts of traffic. In fact, the behaviour of the throughput in figure 3.2.4 proves our point as it has an irregular and inconsistent behaviour as it struggles to maintain a stable throughput rate.

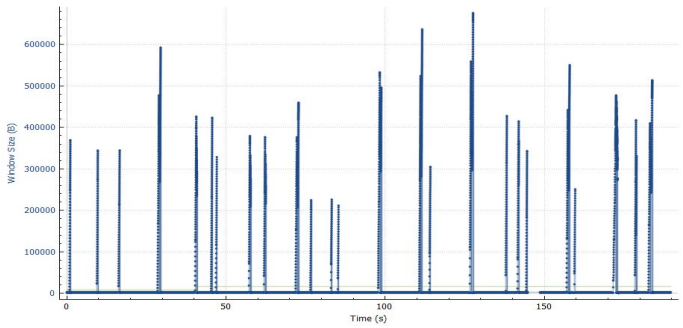

Figure 3.2.4 - Kenya Window Scaling (Server to Receiver)

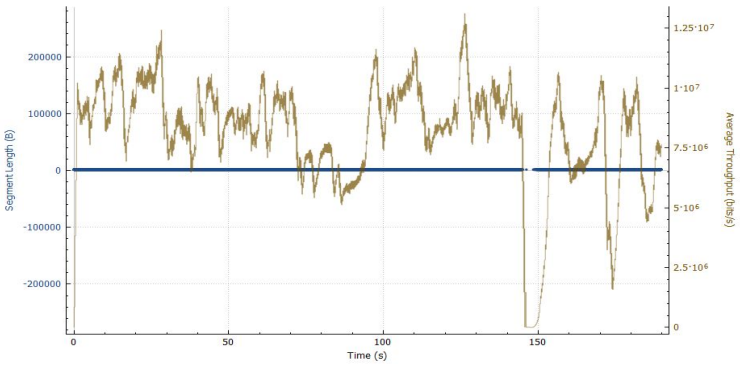

Figure 3.2.5 - Kenya Throughput (bits/s) (Server to Receiver)

Overall, Kenya displays the expected behaviour of the TCP congestion window behaviour, namely the frequency of clear slow-starts and packet losses.

3.3 Lithuania

Below in figure 3.1.1 is our stack trace we found for Lithuania. It demonstrates a fairly consistent throughput, there are no extraordinary features in this trace.

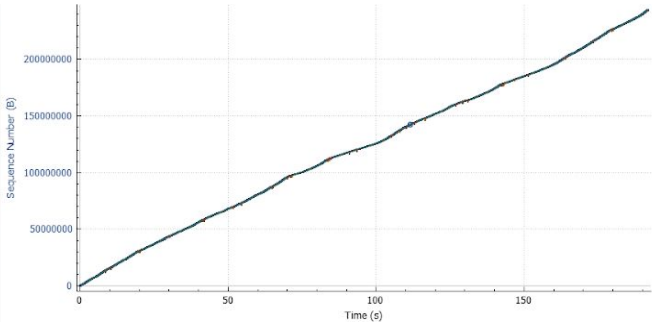

Figure 3.3.1 - Lithuania Sequence Numbers (TCP Trace)

As shown in figure 3.3.3, this trace has a maximum window size of 8269 Bytes. Unlike some of our other traces, this value remains constant throughout the trace and doesn't double as seen in Australia and Kenya. The average throughput of this trace oscillates around 10 million Bits per second, as shown in figure 3.3.4.

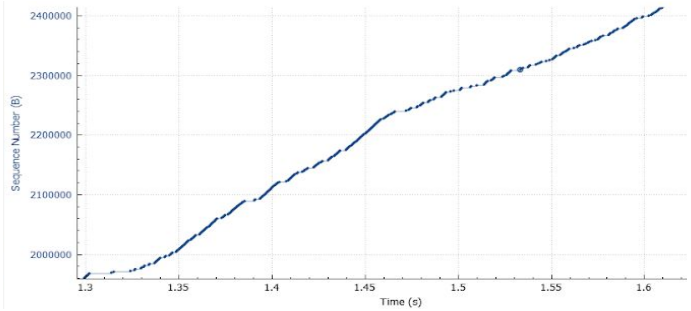

Figure 3.3.2 - Lithuania Slow-Start and Congestion Avoidance Captures

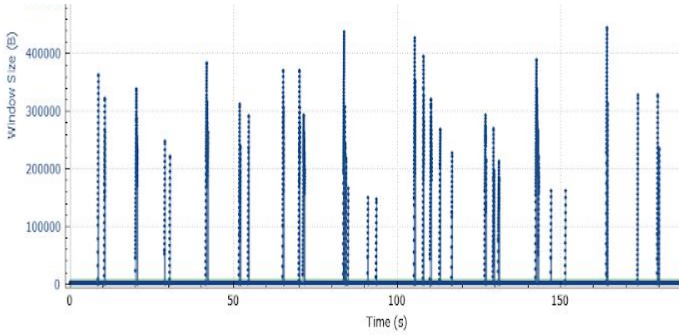

Figure 3.3.3 - Lithuania Window Scaling (Server to Receiver)

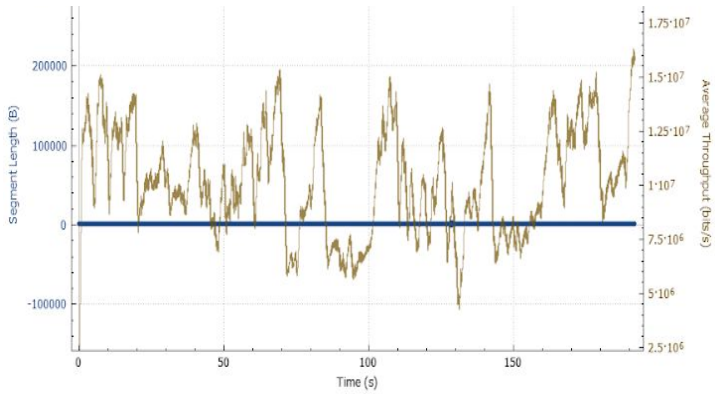

Figure 3.3.4 - Lithuania Throughput (bits/s) (Server to Receiver)

Another important element is the performance of the round trip time (RTT) . In figure 3.3.5, the graph shows a higher density of RTT values below six milliseconds (darker navy), with clear sharp high RTTs when packet losses occur.

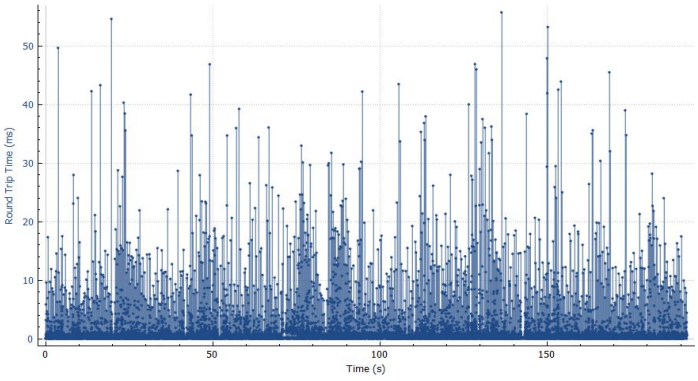

Figure 3.3.5 - Lithuania Round Trip Time (bits/s) (Server to Receiver)

3.4 Slovenia

On a larger scale, the TCP trace for Slovenia in figure 3.4.1 shows no severe events, but rather several small inconsistent bumps. These bumps are indications of slow-start phases due to recovering from packet losses.

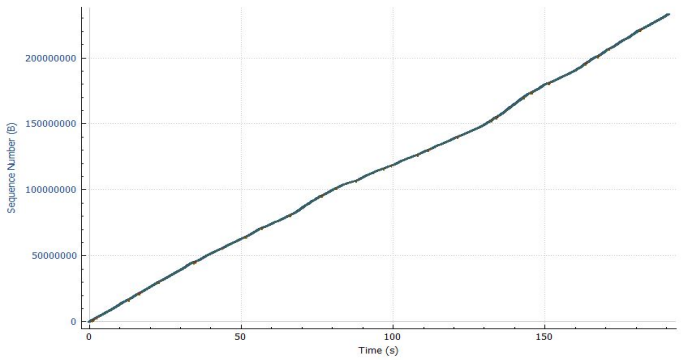

Figure 3.4.1 - Slovenia Sequence Numbers (TCP Trace)

Figure 3.4.2 demonstrates a close-up of the bump at around the 150 second mark from the figure above. This packet loss generated a fast retransmission and we can see that the trace has resumed congestion avoidance stage. Throughout the trace, there were no instances of out-of-order packets.

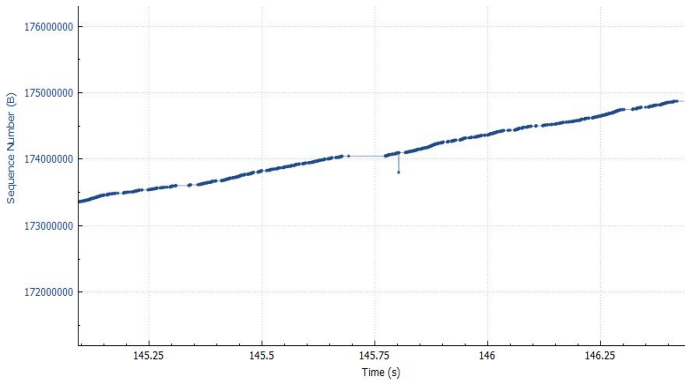

Figure 3.4.2 - Slovenia Fast Retransmission Capture

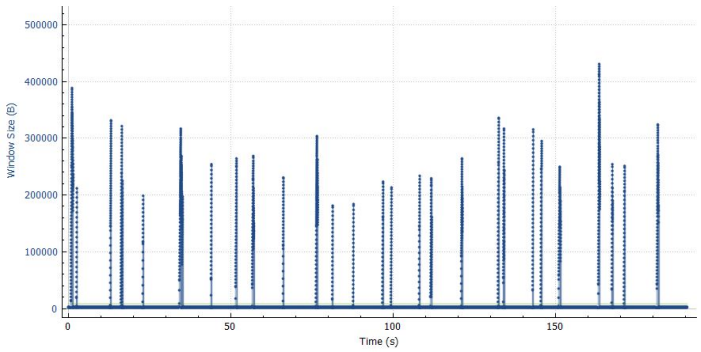

Figure 3.4.3 -Slovenia Window Scaling (Server to Receiver)

From our traces, the maximum congestion window size of the receiver has always remained at 8269 bytes and that the server was sending packets of window size 501 bytes; the highest out of the four traces. One thing to point out is how evenly distributed the frequency of spikes in packet window sizes in figure 3.4.3 compared to the previous countries. We can also see in the below figure that the average throughput oscillates the most and that Slovenia exhibits relatively high RTT, meaning that the transmission-acceptance rate was not optimal.

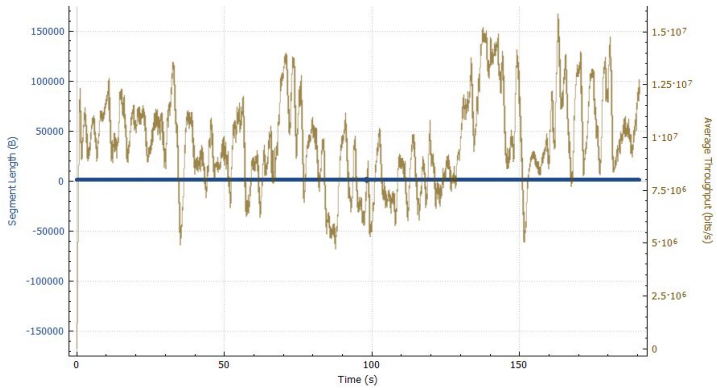

Figure 3.4.4 - Slovenia Throughput (bits/s) (Server to Receiver)

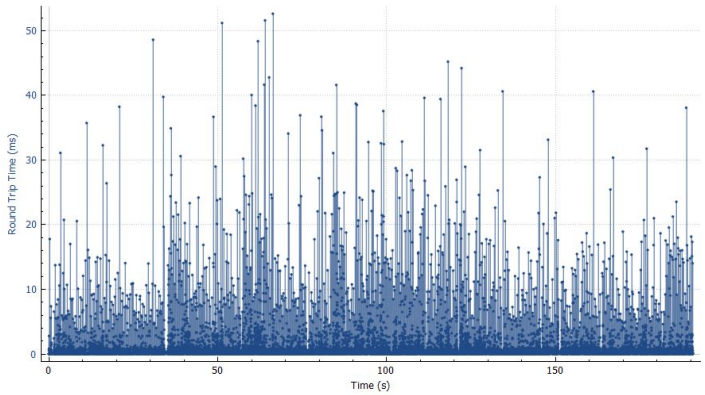

Figure 3.4.5 - Slovenia Round Trip Time (ms) (Server to Receiver)

4. TCP Throughput Analysis

We must use the following equation in order to find the steady state throughputs of our TCP traces. In this equation, R represents steady state throughput, MSS represents maximum segment size, RTT is the average round trip time of the trace, and p is the packet loss rate.

$$R = \frac{MSS}{RTT}\sqrt{\frac{3}{2p}}$$

We found the Average *RTT* by adding RTT as a column in our wireshark, then exported to CSV and found the average of the column.

Country	*Average RTT (s)*
Australia	0.000170517
Kenya	0.000474683
Lithuania	0.0013851
Slovenia	0.000827352

Table 4.1 - Average Round Trip Time per Country

In regards to the *MSS* value, this information can be found on the TCP handshake segments. In our case, the MSS field indicates a value of 1460 byte.

For the loss rate *p*, we found the portion of packet loss by adding the analysis filter: Previous segment(s) not captured (common at capture start) as a column. In this case, the field is blank if there is no packet loss, we counted the number of blanks.

$$p = 1 - \frac{number\ of\ blanks}{number\ of\ total\ packets}$$

Country	*p (%)*
Australia	0.0000486
Kenya	0.000562
Lithuania	0.008252
Slovenia	0.000302

Table 4.2 -Estimated Loss Rate per Country

With the maximum segment size, loss rate and RTT, we are now able to estimate the empirical steady-state throughput (*R*) of the long-duration TCP connections using the calculation below. The results may be found in figure 4.3.

$$R = \frac{MSS}{RTT}\sqrt{\frac{3}{2p}}$$

$$R_A = MSS/(0.000474683) * SQRT(3/2(0.0000486))$$

$$R_A = MSS * 370104.03 = 540,351,883.8 \text{ Bytes/sec}$$

Above is an example of the calculations used to find Steady state Throughput for each of our test-case countries. The results of such calculations on all our test cases can be found below in Table 4.3.

Country	*R (bytes/s)*
Australia	540, 351, 883.8
Kenya	54, 456, 423.2
Lithuania	115, 438, 637.6
Slovenia	124, 366, 873.4

Table 4.3 -Estimated Steady-State Throughput R per Country

5. Discussion

After analyzing traces to explore how different servers with different latency and loss rates affect TCP behaviour, we have deduced that the TCP congestion algorithm heavily depends on the location, distance and internet connection between the sender and receiver. Our main results consist of our ranked steady state throughput. From highest to lowest, our calculations ranked Australia, Slovenia, Lithuania and Kenya. The main challenge behind the logistics of this lab was mainly access to stronger internet connection and equipment. Should we have had access to the equipment and network provided in the McGill Trottier buildings, we would've observed better trends in terms of assessing the maximum congestion window and potentially would have suffered less packet losses. However, we were still able to complete this lab since identifying the causes of packet losses and assessing the RTT, namely from recovering from packet losses, were the main factors.

The suggested equation model for the TCP congestion avoidance algorithm led by the Pittsburgh Supercomputing Center can be used to predict the bandwidth should we have had access to measure the variables more accurately and avoid overestimation as much as possible. However, since the tests were done on a personal connection, it's possible a local delay could have affected our findings. Had we had access to a more substantial and consistent connection, like that of McGill University's, these fluctuations in performance could have had a lesser impact on our findings.

Overall, taking into consideration our analysis and the type of equipment used for this lab, we were still able to confirm that the TCP congestion algorithm heavily depends on the distance between the sender and receiver country, and the strength of the internet connection of both parties and this lab has provide us a solid understanding of TCP behaviour.

References

[1] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," ACM Computer Communication Review, vol 27, no 3, July 1997.